Code:-

Stack.h Header File:-

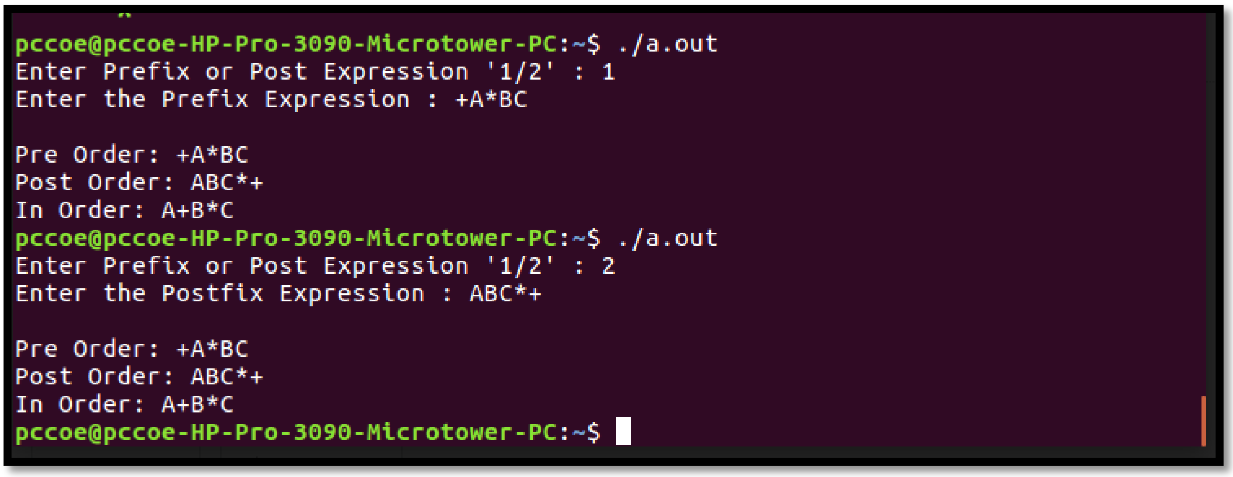
```
#include<stdio.h>
#include<stdlib.h>


typedef struct tree{

char val;
struct tree *lchild;
struct tree *rchild;
}tree;


typedef struct node
{
	tree *data;
	struct node *next;
}stack;

stack* push(stack *top,tree *data)
{
	stack *NewNode=(stack*)malloc(sizeof(stack));
	NewNode->data=data;
	NewNode->next=NULL;
	if(top==NULL)
	{
		top=NewNode;
	}
	else
	{
		NewNode->next=top;
		top=NewNode;
	}
	return top;
}


stack* pop(stack *top)
{
	stack *temp=top;
	if(top==NULL)
	{
		printf("\nStack is Empty");
	}
	else
	{
		top=top->next;
		free(temp);
	}
	return top;
}


tree* ReadTop(stack *top)
{
	return top->data;
}
```

Expression.c File:-

```
#include <stdio.h>
#include "stack.h"
#include <stdlib.h>
#include<string.h>


tree* postfix_ex(tree *root){

		stack *top=NULL;
		char str[30];
		printf("Enter the Postfix Expression : ");
		scanf("%s",&str);

		for(int i=0;str[i]!='\0';i++){

		   tree *tree_node=(tree*)malloc(sizeof(tree));
		   tree_node->val=str[i];
		   tree_node->lchild=tree_node->rchild=NULL;

		   if(isalpha(str[i])){

				top=push(top,tree_node);
		   }
		   else{

				tree_node->rchild=ReadTop(top);
				top=pop(top);
				tree_node->lchild=ReadTop(top);
				top=pop(top);

				top=push(top,tree_node);

		   }
		}

		root=ReadTop(top);
		top=pop(top);
		return root;

}

tree* prefix_ex(tree *root){
        stack *top=NULL;
        char str[30];
        printf("Enter the Prefix Expression : ");
        scanf("%s",&str);
        int length=strlen(str);

        for(int i=length-1;i>=0;i--){

          tree *tree_node=(tree*)malloc(sizeof(tree));
          tree_node->val=str[i];
          tree_node->lchild=tree_node->rchild=NULL;

          if(isalpha(str[i])){

                top=push(top,tree_node);
          }
          else{

                tree_node->lchild=ReadTop(top);
                top=pop(top);
                tree_node->rchild=ReadTop(top);
                top=pop(top);

                top=push(top,tree_node);

          }
        }

        root=ReadTop(top);
        top=pop(top);
        return root;

}


void pre_order(tree *root){

        if(root!=NULL){

          printf("%c",root->val);
          pre_order(root->lchild);
          pre_order(root->rchild);
        }
}

void post_order(tree *root){

        if(root!=NULL){

          post_order(root->lchild);
          post_order(root->rchild);
          printf("%c",root->val);
        }
```

```
}

void in_order(tree *root){
            if(root!=NULL){

               in_order(root->lchild);
               printf("%c",root->val);
               in_order(root->rchild);
            }
}

void main(){

            int ch;
            tree *root=NULL;
            printf("Enter Prefix or Post Expression '1/2'");
            scanf("%d",&ch);

            if(ch==1){
               root=prefix_ex(root);
            }
            else{
               root=postfix_ex(root);
            }

            printf("\nPre Order: ");
            pre_order(root);

            printf("\nPost Order: ");
            post_order(root);

            printf("\nIn Order: ");
            in_order(root);

}
```

Output:-



```
pccoe@pccoe-HP-Pro-3090-Microtower-PC:~$ ./a.out
Enter Prefix or Post Expression '1/2' : 1
Enter the Prefix Expression : +A*BC

Pre Order: +A*BC
Post Order: ABC*+
In Order: A+B*C
pccoe@pccoe-HP-Pro-3090-Microtower-PC:~$ ./a.out
Enter Prefix or Post Expression '1/2' : 2
Enter the Postfix Expression : ABC*+

Pre Order: +A*BC
Post Order: ABC*+
In Order: A+B*C
pccoe@pccoe-HP-Pro-3090-Microtower-PC:~$ 
```